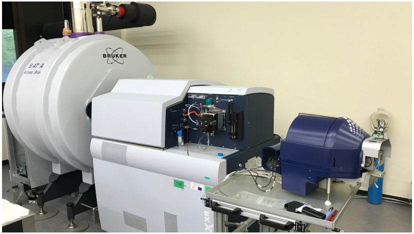
Reconfiguration of graphs with a fixed degree sequence

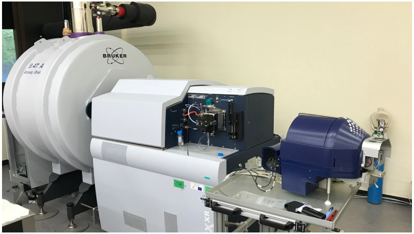
Nicolas Bousquet, Arnaud Mary

WAOA'18

Mass spectrometry

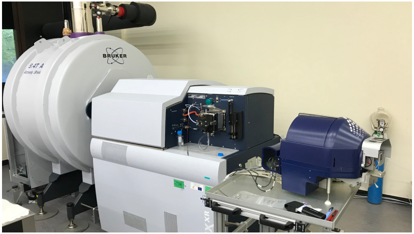


Mass spectrometry

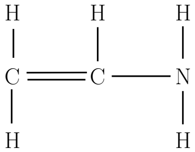


$\Rightarrow$ Chemical formula : C_2NH_5.

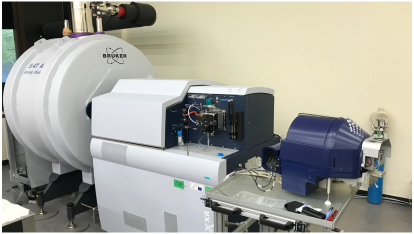
Mass spectrometry



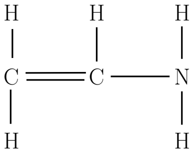
⇒ Chemical formula : C_2NH_5.

⇒ 

Mass spectrometry



$\Rightarrow$ Chemical formula : C_2NH_5.

$\Rightarrow$ 

Molecule = Connected loopless multigraph where

Vertices = Atoms.

Vertex degree = Number of bounds.

Realizing a degree sequence

Mathematical formulation :

Let $S = \{d_1, \ldots, d_n\}$ be a non-increasing sequence. Does it exist a graph satisfying this degree sequence ?

Realizing a degree sequence

Mathematical formulation :
Let $S = \{d_1, \ldots, d_n\}$ be a non-increasing sequence. Does it exist a graph satisfying this degree sequence ?

Theorem ([Senior '51])

Let $S = d_1, \ldots, d_n$ be a non-increasing degree sequence. There exists a connected loop-free multigraph G with degree sequence S iff :

- $\sum d_i$ is even
- $d_n > 0$
- $\sum d_i \geq 2(n-1)$
- $d_1 \leq \sum_{i=2}^{n} d_i$.

Realizing a degree sequence

Mathematical formulation :

Let $S = \{d_1, \ldots, d_n\}$ be a non-increasing sequence. Does it exist a graph satisfying this degree sequence ?

Theorem ([Senior '51])

Let $S = d_1, \ldots, d_n$ be a non-increasing degree sequence. There exists a connected loop-free multigraph G with degree sequence S iff :

- $\sum d_i$ is even
- $d_n > 0$
- $\sum d_i \geq 2(n-1)$
- $d_1 \leq \sum_{i=2}^{n} d_i$.

Question :

Is it necessarily the correct molecule ? $\Rightarrow$ NO !

Structural isomers

Two molecules can have the same degree sequence, they are called (structural) isomers.

Structural isomers

Two molecules can have the same degree sequence, they are called (structural) isomers.

Question :
Is it possible to generate (efficiently) all the molecules with a fixed degree sequence ?

Structural isomers

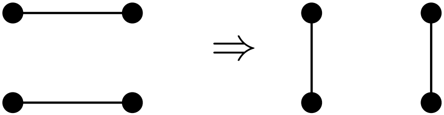
Two molecules can have the same degree sequence, they are called (structural) isomers.

Question :
Is it possible to generate (efficiently) all the molecules with a fixed degree sequence ?

Generation from a seed :

- We start from a graph G with a fixed degree sequence
- We apply an operation that maintains the degree sequence.
- Generation of all the graphs of that DS by repeating this operation ?

The natural operation : flip



Reconfiguration graph

Given a degree sequence S, $\mathcal{G}(S)$ is the graph where

- Vertices = loopless multigraphs with degree sequence S.
- Edge G_1, G_2 = There is a flip transforming G_1 into G_2.

Reconfiguration graph

Given a degree sequence S, $\mathcal{G}(S)$ is the graph where

- Vertices = loopless multigraphs with degree sequence S.
- Edge G_1, G_2 = There is a flip transforming G_1 into G_2.

Remark :
Any loopless multigraph G_1 with degree sequence S can be transformed into G_2 via flips
$\Leftrightarrow \mathcal{G}(S)$ is connected.

Reconfiguration graph

Given a degree sequence S, $\mathcal{G}(S)$ is the graph where

- Vertices = loopless multigraphs with degree sequence S.
- Edge G_1, G_2 = There is a flip transforming G_1 into G_2.

Remark :
Any loopless multigraph G_1 with degree sequence S can be transformed into G_2 via flips
$\Leftrightarrow$ $\mathcal{G}(S)$ is connected.

Restriction of the reconfiguration graph :
Given a property Π, we denote by $\mathcal{G}(S, \Pi)$ the induced subgraph of $\mathcal{G}(S)$ restricted to graphs with property Π.

Classical properties Π : Connected, being simple...etc...

Existing results

- Find a graph with a fixed degree sequence S if it exists ?
- Generate all the graphs of degree sequence S using flips ?
- Given two graphs can we find a shortest transformation ?
- Given two graphs can we approximate a shortest transformation ?

Existing results

- Find a graph with a fixed degree sequence S if it exists ?
  Polytime [Hakimi '62]
- Generate all the graphs of degree sequence S using flips ?
  YES [Hakimi '62]
- Given two graphs can we find a shortest transformation ?
  NP-complete [Will '99]
- Given two graphs can we approximate a shortest transformation ?
  3/2-approx [Bereg, Ito '17]

Multigraphs

Existing results

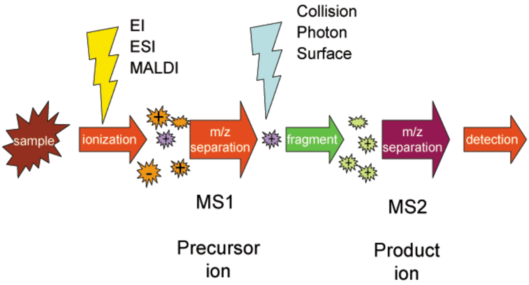
- Find a graph with a fixed degree sequence S if it exists ?
  Polytime [Hakimi '62] [Senior '51]
- Generate all the graphs of degree sequence S using flips ?
  YES [Hakimi '62] [Taylor '81]
- Given two graphs can we find a shortest transformation ?
  NP-complete [Will '99] [B., Mary '18]
- Given two graphs can we approximate a shortest transformation ?
  3/2-approx [Bereg, Ito '17] 4-approx [B., Mary '18]

Multigraphs Connected multigraphs.

Tandem mass spectrometry

Figure : wikipedia.com

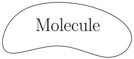
Tandem mass spectrometry



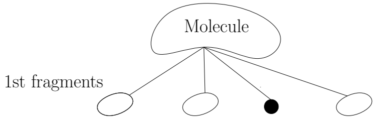
- The molecule is broken into pieces...
- ... which is in turn again broken into pieces...etc...

Figure : wikipedia.com

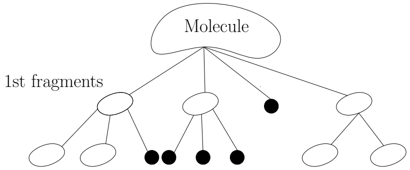
Tree of the fragments



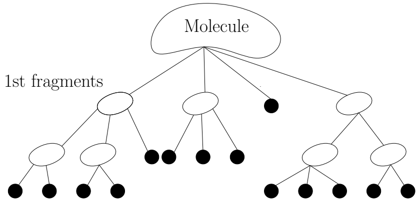
Tree of the fragments



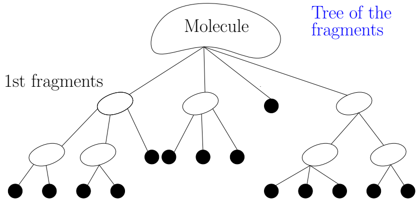
Tree of the fragments



Tree of the fragments



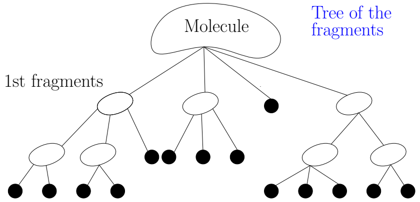
Tree of the fragments



Tree of the fragments :

- Each leaf is an atom $\rightarrow$ its degree is known.

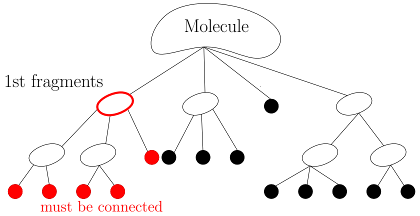
Tree of the fragments



Tree of the fragments :

- Each leaf is an atom $\rightarrow$ its degree is known.
- The whole graph is connected.

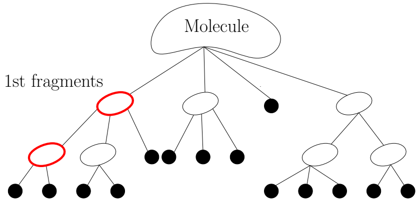
Tree of the fragments



Tree of the fragments :

- Each leaf is an atom → its degree is known.
- The whole graph is connected.
- Each fragment induces a connected subgraph.

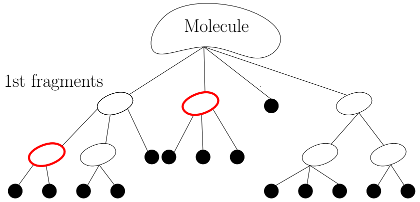
Tree of the fragments



Tree of the fragments :

- Each leaf is an atom → its degree is known.
- The whole graph is connected.
- Each fragment induces a connected subgraph.
- Fragments are pairwise included in the other or disjoint → the collection of fragments is nested.

Tree of the fragments



Tree of the fragments :

- Each leaf is an atom → its degree is known.
- The whole graph is connected.
- Each fragment induces a connected subgraph.
- Fragments are pairwise included in the other or disjoint → the collection of fragments is nested.

Combinatorial reformulation

A degree sequence S.
A set of fragments $\mathcal{C}$ that

- contains V
- is nested.

Our property Π :
For every $C \in \mathcal{C}$, $G[C]$ is connected.

$\mathcal{G}(S, \Pi)$: graphs of $\mathcal{G}(S)$ such that every set in $\mathcal{C}$ induces a connected subgraph.

Combinatorial reformulation

A degree sequence S.
A set of fragments $\mathcal{C}$ that

- contains V
- is nested.

Our property Π :
For every $C \in \mathcal{C}$, $G[C]$ is connected.

$\mathcal{G}(S,\Pi)$: graphs of $\mathcal{G}(S)$ such that every set in $\mathcal{C}$ induces a connected subgraph.

Questions : Can we still :

- Find a graph that realizes this degree sequence S where each set of $\mathcal{C}$ is connected ?
  $\Leftrightarrow$ Find a graph in $\mathcal{G}(S,\Pi)$?
- Generate all the solutions using flips ?
  $\Leftarrow$ Is $\mathcal{G}(S,\Pi)$ connected ?

Our results

Theorem (B., Mary)

We can find in polynomial time a graph in $\mathcal{G}(S, \Pi)$ if it exists.

Theorem (B., Mary)

$\mathcal{G}(S, \Pi)$ is connected.

The proof is algorithmic and we can moreover prove the following :

Theorem B., Mary)

Given G_1, G_2 in $\mathcal{G}(S, \Pi)$, we can find in polynomial time a transformation from G_1 to G_2 of length at most $(8d + 4) \cdot OPT$.

Our results

Theorem (B., Mary)

We can find in polynomial time a graph in $\mathcal{G}(S, \Pi)$ if it exists.

Theorem (B., Mary)

$\mathcal{G}(S, \Pi)$ is connected.

The proof is algorithmic and we can moreover prove the following :

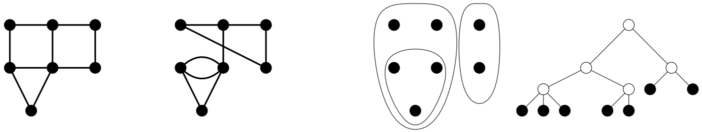
Theorem B., Mary)

Given G_1, G_2 in $\mathcal{G}(S, \Pi)$, we can find in polynomial time a transformation from G_1 to G_2 of length at most $(8d + 4) \cdot OPT$.

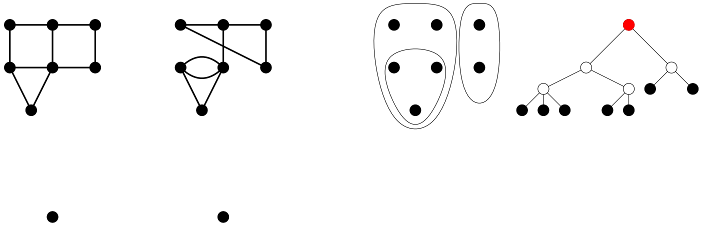
Corollary :

There is a polynomial delay algorithm to enumerate all the graphs in $\mathcal{G}(S, \Pi)$.

Tree augmentation

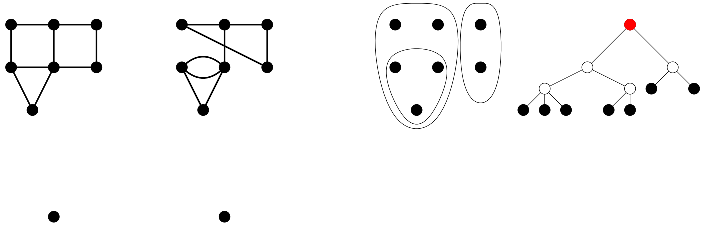


Tree augmentation



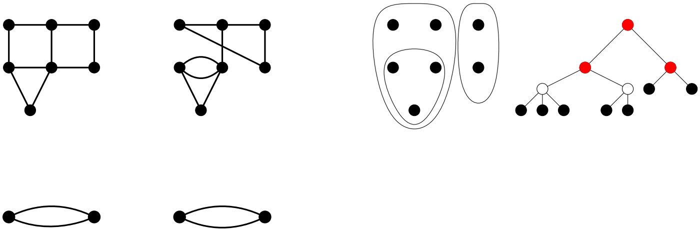
- Start from the root of the tree of the fragments.
- Auxiliary graph : all the fragments that are leaves of the current tree are contracted.

Tree augmentation



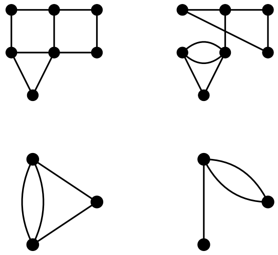
- Start from the root of the tree of the fragments.
- Auxiliary graph : all the fragments that are leaves of the current tree are contracted.
- If the two auxiliary graphs agree, add all the children of a leaf of a current tree.

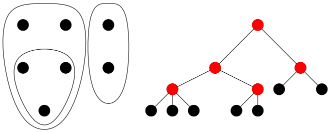
Tree augmentation



- Start from the root of the tree of the fragments.
- Auxiliary graph : all the fragments that are leaves of the current tree are contracted.
- If the two auxiliary graphs agree, add all the children of a leaf of a current tree.

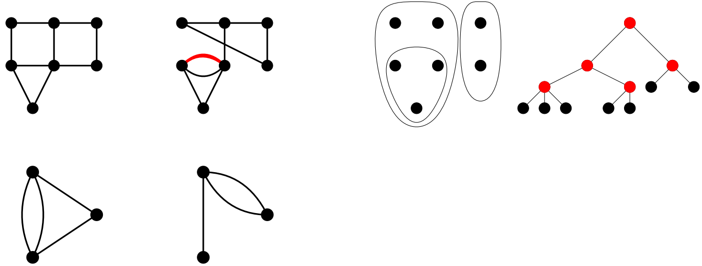
Tree augmentation





- Start from the root of the tree of the fragments.
- Auxiliary graph : all the fragments that are leaves of the current tree are contracted.
- If the two auxiliary graphs agree, add all the children of a leaf of a current tree.

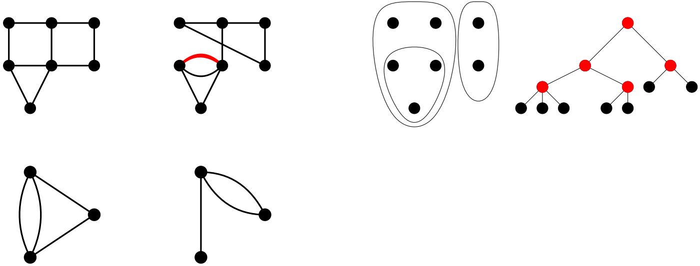
Step 1 : Equilibration the degree sequences



Claim :

If C has larger degree in the auxiliary graph of G than in H then an edge of $H[C]$ can be deleted without violating any constraints.

Step 1 : Equilibration the degree sequences



Claim :

If C has larger degree in the auxiliary graph of G than in H then an edge of $H[C]$ can be deleted without violating any constraints.

Sketch :

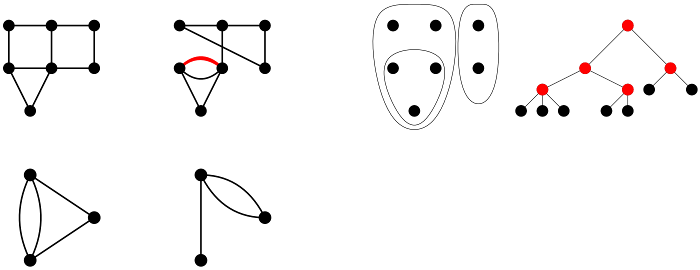
Same S + Assumptions $\Rightarrow E(H[C]) > E(G[C])$.

$\Rightarrow H[C]$ contains a (nice) cycle D.

$\Rightarrow$ Delete an edge of D does not violate connectivity constraints.

semi false...

Step 1 : Equilibration the degree sequences



Claim :

If C has larger degree in the auxiliary graph of G than in H then an edge of $H[C]$ can be deleted without violating any constraints.

Sketch :

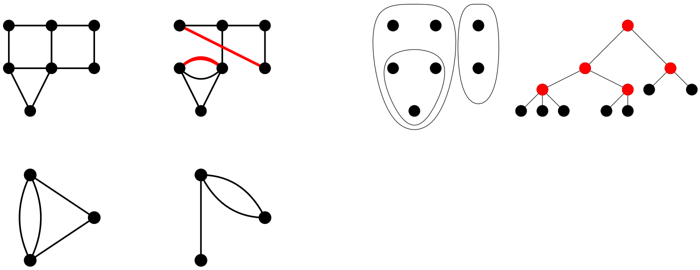
Same S + Assumptions $\Rightarrow E(H[C]) > E(G[C])$.

$\Rightarrow H[C]$ contains a (nice) cycle D.

$\Rightarrow$ Delete an edge of D does not violate connectivity constraints.

semi false...

Step 1 : Equilibration the degree sequences



Claim :

If C has larger degree in the auxiliary graph of G than in H then an edge of $H[C]$ can be deleted without violating any constraints.

Sketch :

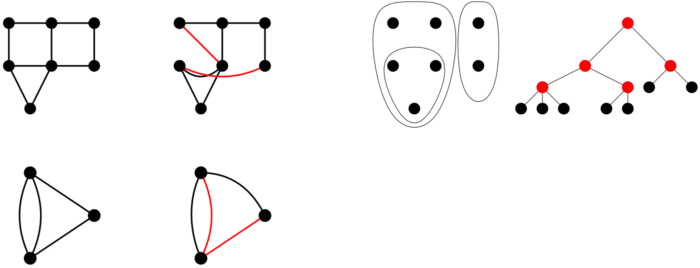
Same S + Assumptions $\Rightarrow E(H[C]) > E(G[C])$.

$\Rightarrow H[C]$ contains a (nice) cycle D.

$\Rightarrow$ Delete an edge of D does not violate connectivity constraints.

semi false...

Step 1 : Equilibration the degree sequences



Claim :

If C has larger degree in the auxiliary graph of G than in H then an edge of $H[C]$ can be deleted without violating any constraints.

Sketch :

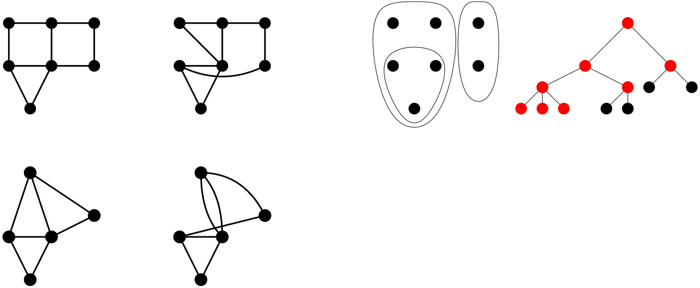
Same S + Assumptions $\Rightarrow E(H[C]) > E(G[C])$.

$\Rightarrow H[C]$ contains a (nice) cycle D.

$\Rightarrow$ Delete an edge of D does not violate connectivity constraints.

semi false...

Step 1 : Equilibration the degree sequences

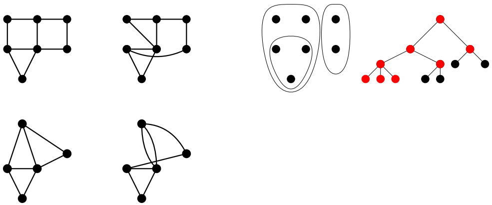


Claim :
If C has larger degree in the auxiliary graph of G than in H then an edge of $H[C]$ can be deleted without violating any constraints.

Sketch :
Same S + Assumptions $\Rightarrow E(H[C]) > E(G[C])$.
$\Rightarrow H[C]$ contains a (nice) cycle D.
$\Rightarrow$ Delete an edge of D does not violate connectivity constraints.

semi false...

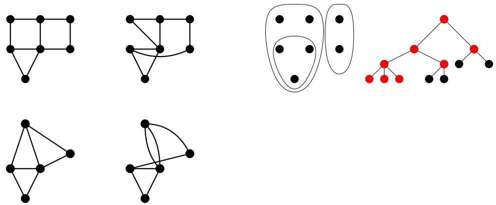
Step 2 : Transform the auxiliary graph



Theorem ([Taylor] [B., Mary])

It is possible to transform the first reduced graph into the second maintaining connectivity.

Step 2 : Transform the auxiliary graph

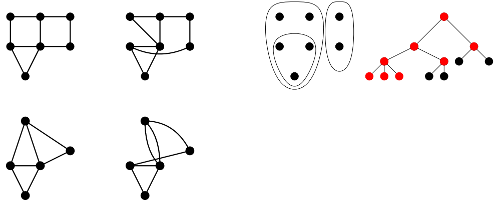


Theorem ([Taylor] [B., Mary])

It is possible to transform the first reduced graph into the second maintaining connectivity.

Problem : Not enough ! (some subsets have to remain connected).

Step 2 : Transform the auxiliary graph



Theorem ([Taylor] [B., Mary])

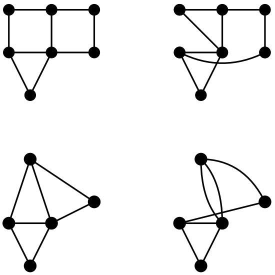
It is possible to transform the first reduced graph into the second maintaining connectivity.

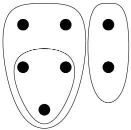
Problem : Not enough ! (some subsets have to remain connected).

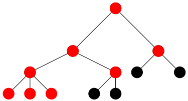
Since the graphs agree before the "extension", the difference is "reduced" to the new set of vertices.

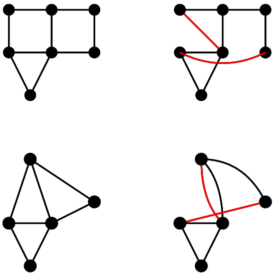
⇒ Only modify edges incident to new vertices.

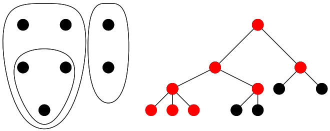
⇒ To ensure it, we define a (new !) auxiliary graph.

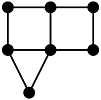


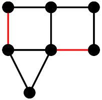


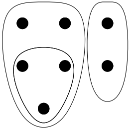


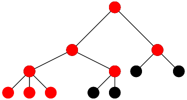


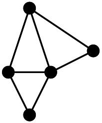


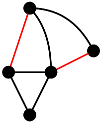


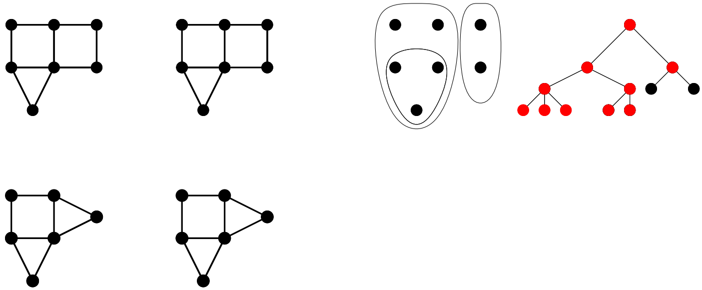




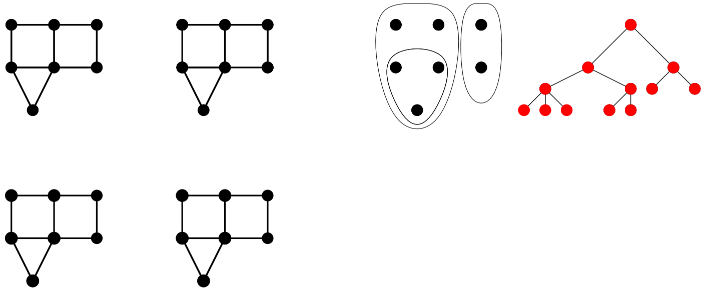








Current tree = Whole tree of the fragments ⇒ the graphs are the same.



Current tree = Whole tree of the fragments ⇒ the graphs are the same.

Approximation ratio

Claim 1

We never flip a good edge (an edge of both graphs).

Claim 2

An edge of the symmetric is used at most once to equilibrate degrees.

Claim 3

An edge is modified only when one of its endpoints is “extended” in the tree.

$\Rightarrow$ $(8d + 4)$-approximation algorithm.

Conclusion

- Improve the approximation factor. Can we eliminate the factor depending of the height ?

Conclusion

- Improve the approximation factor. Can we eliminate the factor depending of the height ?
- For simple graphs.
  Best approximation factor : $\frac{3}{2}$. No lower bound.
- For connected graphs.
  Best approximation factor : 4. No lower bound.

Conclusion

- Improve the approximation factor. Can we eliminate the factor depending of the height ?
- For simple graphs.
  Best approximation factor : $\frac{3}{2}$. No lower bound.
- For connected graphs.
  Best approximation factor : 4. No lower bound.
- What if the collection $\mathcal{C}$ is not nested ?

Conclusion

- Improve the approximation factor. Can we eliminate the factor depending of the height ?
- For simple graphs.
  Best approximation factor : $\frac{3}{2}$. No lower bound.
- For connected graphs.
  Best approximation factor : 4. No lower bound.
- What if the collection $\mathcal{C}$ is not nested ?
- Polynomial space enumeration algorithm with polynomial delay ?

Conclusion

- Improve the approximation factor. Can we eliminate the factor depending of the height ?
- For simple graphs.
  Best approximation factor : $\frac{3}{2}$. No lower bound.
- For connected graphs.
  Best approximation factor : 4. No lower bound.
- What if the collection $\mathcal{C}$ is not nested ?
- Polynomial space enumeration algorithm with polynomial delay ?

Thanks for your attention